# *"NewSQL:" C-Store, Spanner*

# *Where We've Been …*

- an interesting recent system
- C-Store (commercial fork: Vertica)
  - Store tables by columns rather than rows to optimize performance for OLAP queries

# *Where We're Going …*

- Two more interesting recent systems
- H-Store (commercial fork: VoltDB)
  - Custom architecture tuned for OLTP workloads
  - Heavy use of main memory
- Spanner (an internal system used at Google)
  - SQL-like language over a key-value store
  - Strong consistency (Paxos!!)

# H-Store/VoltDB

- What about OLTP workloads?
- We can improve performance on these too!
- Exploit modern hardware
- Exploit unique features of OLTP workloads
- Example: H-Store (comercialized as VoltDB)

# *Today's OLTP workloads*

- Data is relatively small, < 100 GB typically
- Transactions are short-running, no "user stalls"
  - This is no longer the 1970s where you input SQL at a terminal
  - When you buy something on Amazon, typically split into several underlying transactions

# *Today's computers*

- Memory is no longer tiny
  - 100GB of RAM? Sure, you can outfit a machine with that
- Your database no longer sits on a single box
  - Have available infrastructure that is distributed and replicated for fault-tolerance

# H-Store design decisions

- Run everything in memory
  - Could rely on replication and failover for durability
    - So, only need to log for undo purposes (not for redo)
  - Though VoltDB does use periodic disk snapshots

# H-Store design decisions

- Run all transactions serially
  - Hey, they're short anyway
- Result: saves on some major overhead
  - Disk accesses
  - Synchronization/concurrency

# *OLTP Workloads*

- Make use of the fact that OLTP workloads are not ad-hoc
- Require all possible transaction classes to be predefined and registered with the system
  - Can be pre-optimized
  - For distributed transactions, can identify which of them really require inter-site communication/2PC
- Allows a better DB design as we know the entire workload up front (data partitioning etc.)

# *Summary so far*

- Custom solutions and architectures for particular classes of applications
- Column stores for read-mostly, OLAP style workloads
- H-Store and similar systems for OLTP workloads
  - In-memory
  - Transactions run serially
  - Optimized for a fully pre-specified workload

# Google Spanner

- Distributed multiversion database
  - General-purpose transactions (ACID)
  - SQL query language
  - Schematized tables
  - Semi-relational data model
- Running in production
  - Storage for Google's ad data
- Presented at OSDI (major systems conference) in 2012

# Overview

- Supports lock-free distributed read transactions using snapshots
- Guarantees external consistency (linearizability) of distributed transactions
  - A combination of serializability and linearizability
  - If T1 commits before T2 starts, then T1 is serialized before T2
  - First system at global scale to enforce this

# Read Transactions

- In a social network, generate a page of friends' recent posts
  - Consistent view of friend list and their posts

Why consistency matters

1. Remove untrustworthy person X as friend
2. Post P: "My government is repressive…"

# Synchronizing snapshots

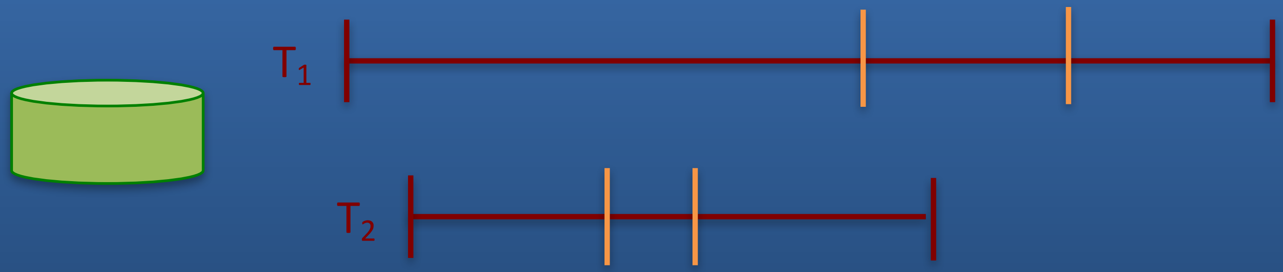
- Read snapshots must be consistent across multiple sites
- Implementation relies on appropriate use of transaction <u>timestamps</u>
  - <u>In distributed commit!</u>

# Assigning Timestamps

- Strict two-phase locking for write transactions
- Assign timestamp while locks are held

# Some Timestamp Guarantees

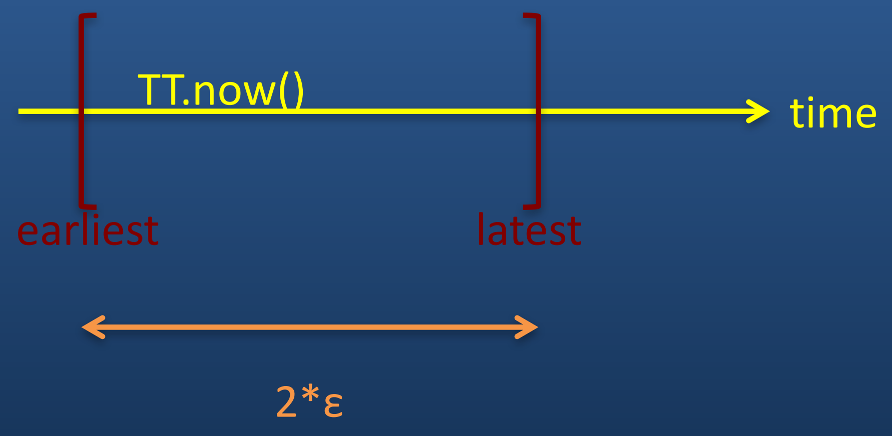
- For conflicting transactions, timestamp order == serialization order



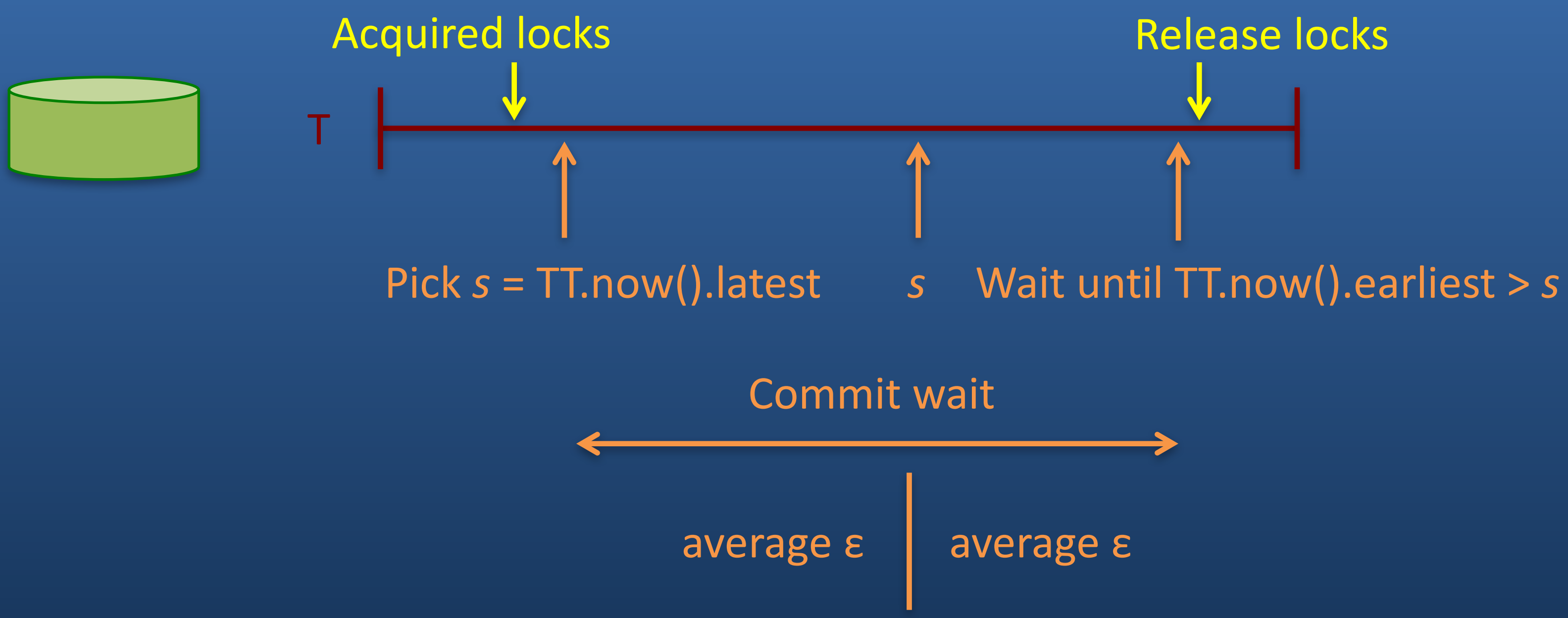

- T4 starts after T3 ends => T4 has larger timestamp

# TrueTime API

- “Global wall-clock time” with bounded uncertainty



# Timestamps and TrueTime



# Distributed Timestamps w/2PC

- Coord sends Prepare
- Sub replies Yes(~ TT.now().latest)
- Coord computes s =
  max ({ replies }, TT.now().latest)
- Coord waits until TT.now.earliest > s
- Coord sends Commit(s)
- Coord waits for ACKs

# Subtle Issues

- Message propagation delay
- Sub reply must be > any previous proposal by that sub
- Reader with snapshot greater than some outstanding proposal must block

# Summary

- Lock-free read transactions across datacenters
- External consistency
  - A very strong formal guarantee
- TrueTime
  - Uncertainty in time can be waited out
- More details (e.g. how to actually implement consistent reads at a time/version) in paper

# *Slide credits*

- VLDB 2009 tutorial on Column stores
  - http://www.cs.yale.edu/homes/dna/talks/Column_Store_Tutorial_VLDB09.pdf
- H-Store slides
  - http://hstore.cs.brown.edu/slides/hstore-vldb2007.pdf
- Google Spanner Slides
  - http://research.google.com/archive/spanner-osdi2012.pptx (substantially modified)